NOTE: Use these buttons to print or save the form. DO NOT use the browser tool bar. SAVE PRINT

LA-UR-10-08035

Approved for public release;
distribution is unlimited.

Title: Data-Intensive Computing on Numerically-Intensive Supercomputers

Author(s): Ahrens James P. 113788 CCS-7, Fasel Patricia K. 090207 CCS-3, Habib Salman 109589 T-2, Heitmann Katrin 175878 ISR-1, Hsu Chung-Hsing Oakridge National Laboratory, Lo Li-Ta 194699 CCS-7, Patchett John M. 148176 CCS-7, Williams Sean J. 230347 CCS-7, Woodring Jonathan L. 209118 CCS-7, Wu Joshua 233306 CCS-7

Intended for: 2010 Super Computing Conference Nov. 2010

Los Alamos NATIONAL LABORATORY EST.1943

Form 836 (7/06)

SAVE PRINT CLEAR FORM

Data-Intensive Analysis and Visualization on Numerically-Intensive Supercomputers

Abstract: With the advent of the era of petascale supercomputing, via the delivery of the Roadrunner supercomputing platform at Los Alamos National Laboratory, there is a pressing need to address the problem of visualizing massive petascale-sized results. In this presentation, I discuss progress on a number of approaches including in-situ analysis, multi-resolution out-of-core streaming and interactive rendering on the supercomputing platform. These approaches are placed in context by the emerging area of data-intensive supercomputing.

Bio: James Ahrens graduated with his Ph.D. in Computer Science from the University of Washington. His dissertation topic was on a high-performance scientific visualization and experiment management system. After graduation he joined Los Alamos National Laboratory as a staff member working for the Advanced Computing Laboratory (ACL). He is currently the visualization team leader in the ACL. His research areas of interest include methods for visualizing extremely large scientific datasets, distance visualization and quantitative/comparative visualization.

Data-Intensive Computing on Numerically-Intensive Supercomputers

James Ahrens
Los Alamos National Laboratory

Patricia Fasel, Salman Habib, Katrin Heitmann, Chung-Hsing Hsu, Ollie Lo, John Patchett, Sean Williams, Jonathan Woodring, Joshua Wu

November 2010

A "Middle Way" between:

- Numerically-intensive / HPC approach
  - Massive FLOPS
    - Top 500 list – 1999 Terascale, 2009 Petascale, 2019? Exascale
    - Roadrunner – First petaflop supercomputer – Opteron, Cell
- Data-intensive supercomputing (DISC) approach
  - Massive data

- We are exploring it by necessity for interactive scientific visualization of massive data
  - DISC using a traditional HPC platform

Trends in supercomputing drive visualization solutions

Prefix	Mega	Giga	Tera	Peta	Exa
10^n	10^6	10^9	10^{12}	10^{15}	10^{18}
Technology	Displays, networks →		Data sizes and machines →		

What is Data Intensive Supercomputing?

- Data Intensive Super Computing (DISC)
  - Definition by Randal Byrant, CMU
  - 1. Data as first-class citizen
  - 2. High-level data oriented programming model
  - 3. Interactive access – human in the loop
  - 4. Reliability
- Large database community driver
  - Success of Google's map reduce approach
    - Hundred of processors, terabytes of data, tenth of second response time
- Scientific driver
  - Massive data from simulations, experiments, observations
- DISC highlights the downsides of pursuing a straight massive FLOPS approach

Outline of presentation

- Explore the “Middle way” through HPC/DISC using real-world examples from scientific visualization
  - Use DISC as a topic guide
- 1. Data as first-class citizen
  - In-situ analysis for Roadrunner Universe application
- 2. High-level data-oriented programming model
  - Programming visualization tools
  - Multi-resolution out-of-core visualization
- 3. Interactive access – human in the loop
  - Visualization on the supercomputing platform
- 4. Reliability

1. Data as a first class citizen – Data storage is the major architectural difference between HPC and DISC

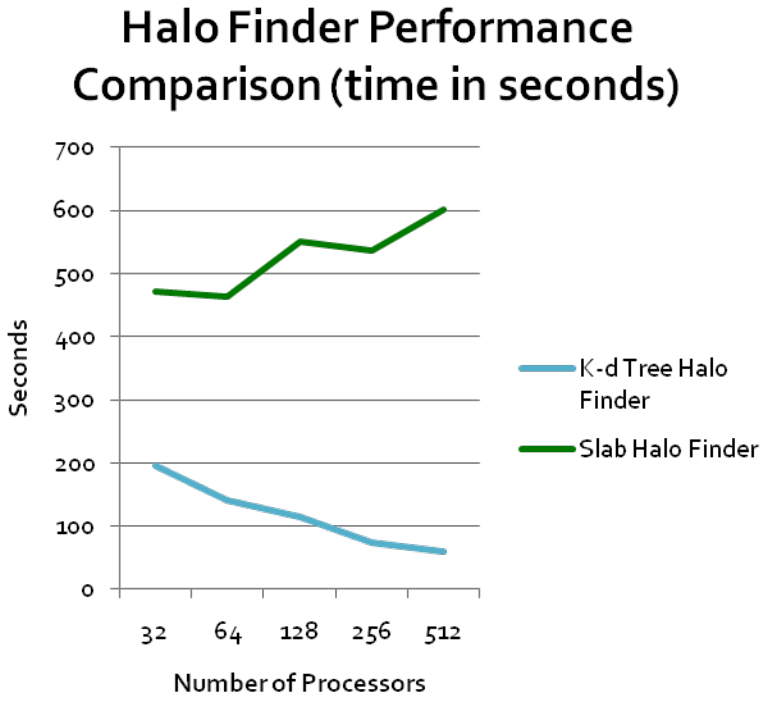
- Numerically-intensive
  - Data stored in parallel filesystem
  - Brought into system for computation
- Data-intensive
  - Computation co-located with storage
- Numerically-intensive / Roadrunner example
  - Petaflop supercomputer with a few petabytes of disk
- Think hard about a data-focused approach (data first!) on a numerically-intensive supercomputer
  - What specific scientific questions will this petascale run answer? With what data?
  - What are the algorithms to do this?

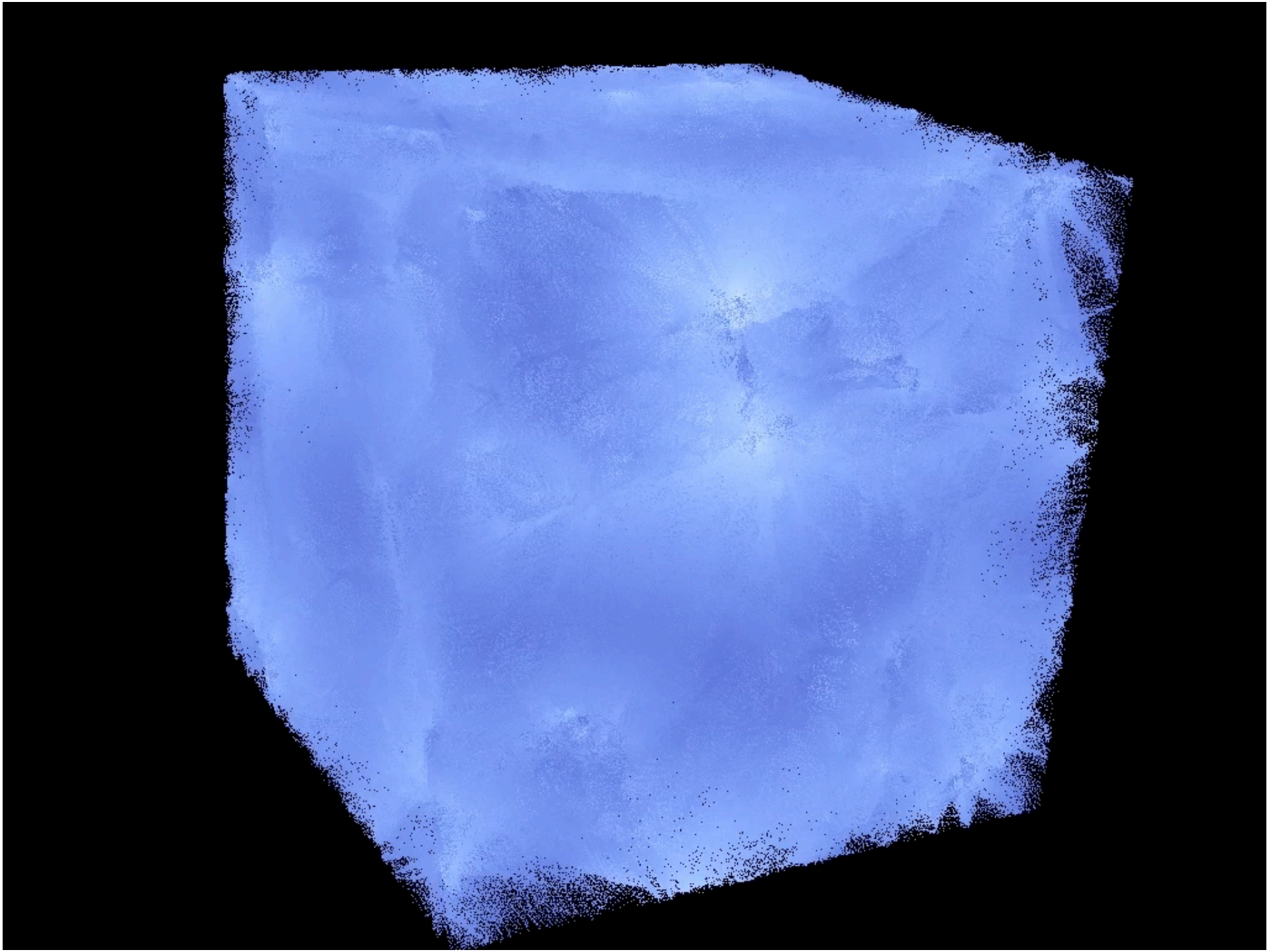
Data as a first class citizen in the Roadrunner Universe project (RRU)

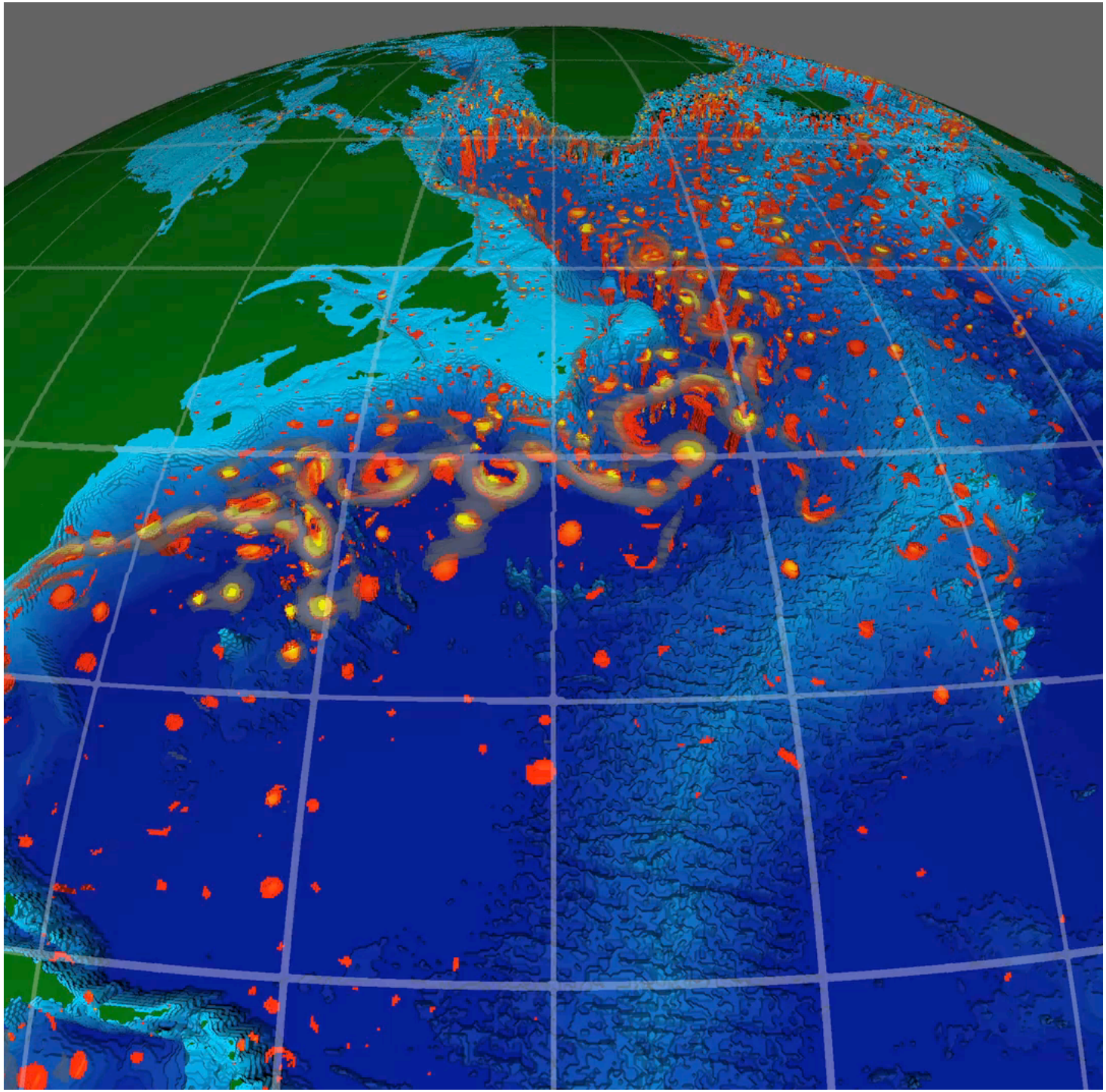
- RRU -- First petascale cosmology simulations
  - New scalable hybrid code designed for heterogeneous architectures
  - New algorithmic ideas for high performance
    - Domain overloading with particle caches
    - Digital filtering to reduce communication across Opteron/Cell layer
    - >50 times speed-up over conventional codes
- RRU data challenge
  - Individual trillion particle runs generate 100s of TB of raw data
- Must carry out “on the fly” analysis
  - KD tree-based halo finder parallelized with particle overloading

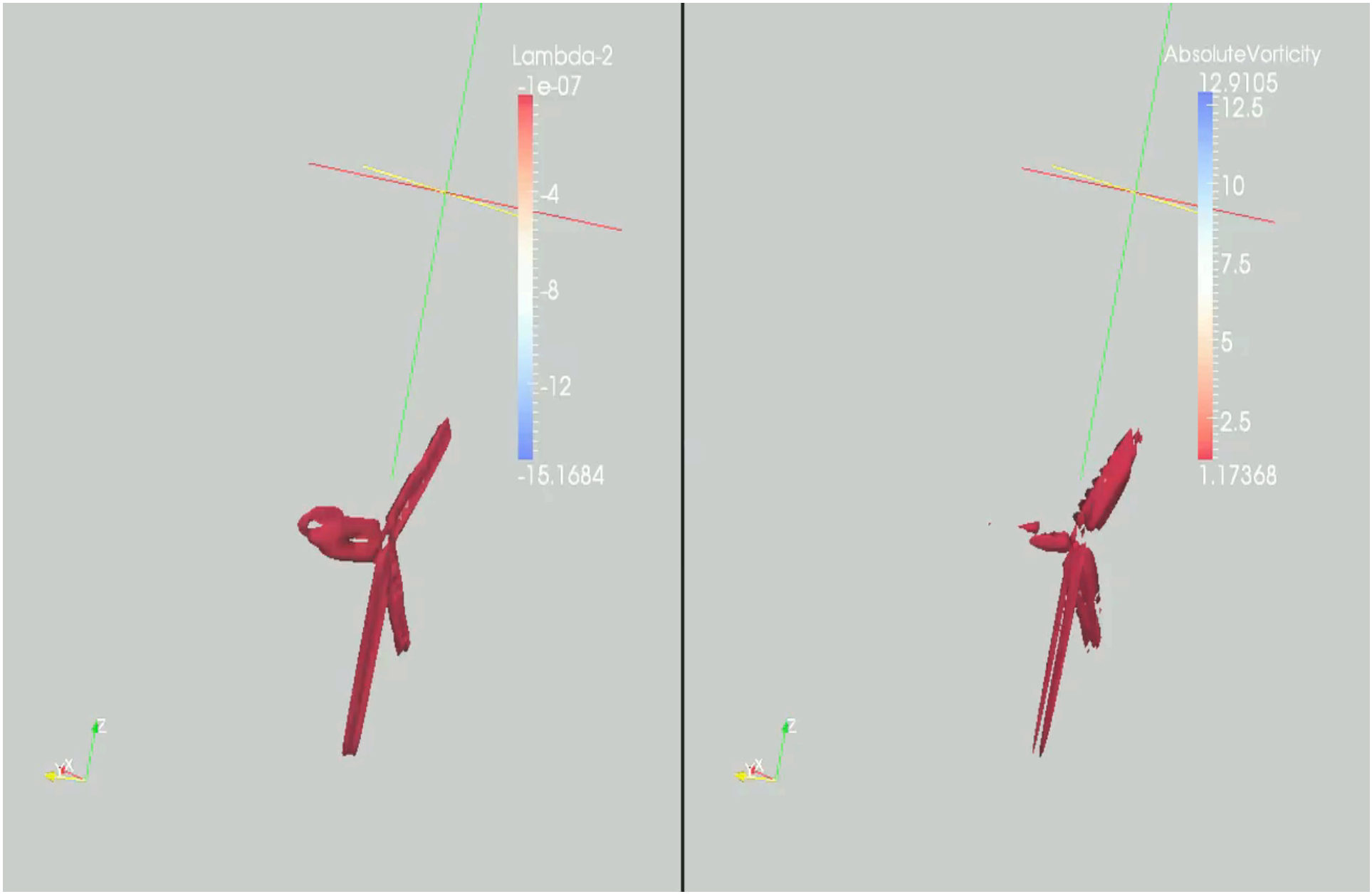
Impact: Science at massive scale requires efficient and effective data analysis

- Data reduction through in-situ feature extraction
  - Save every hundredth halo catalog
    - Every output timestep, save properties and statistics of halo
  - Optimized performance



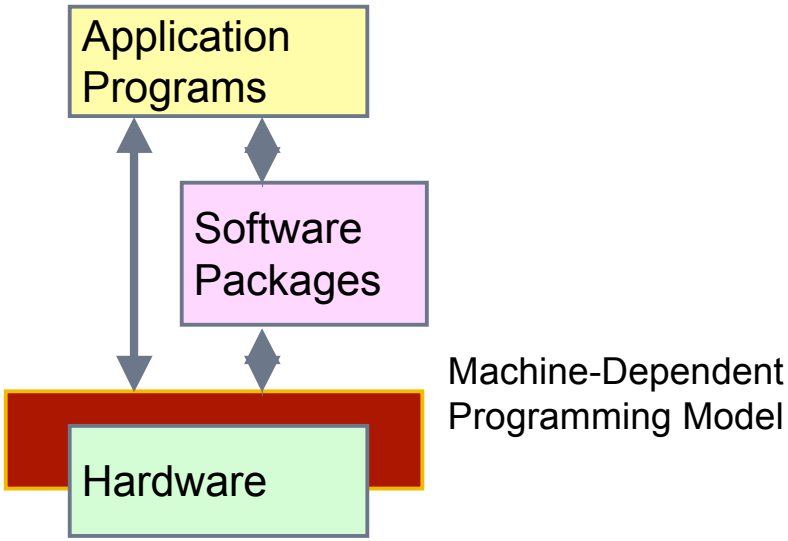





Lambda-2
-1e-07
-4
-8
-12
-15.1684
Z
X
AbsoluteVorticity
12.9105
12.5
10
7.5
5
2.5
1.17368
Z
X

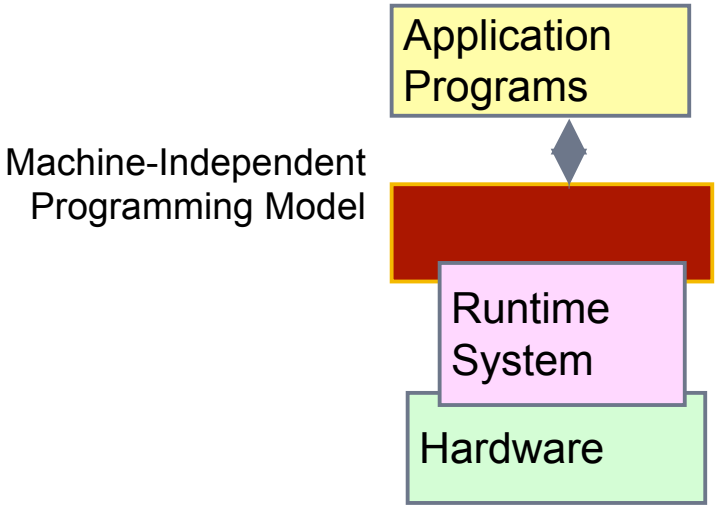
2. DISC requires a high-level data-oriented programming model

Numerically Intensive



- Programs described at very low level (MPI)
- Rely on small number of software packages

DISC

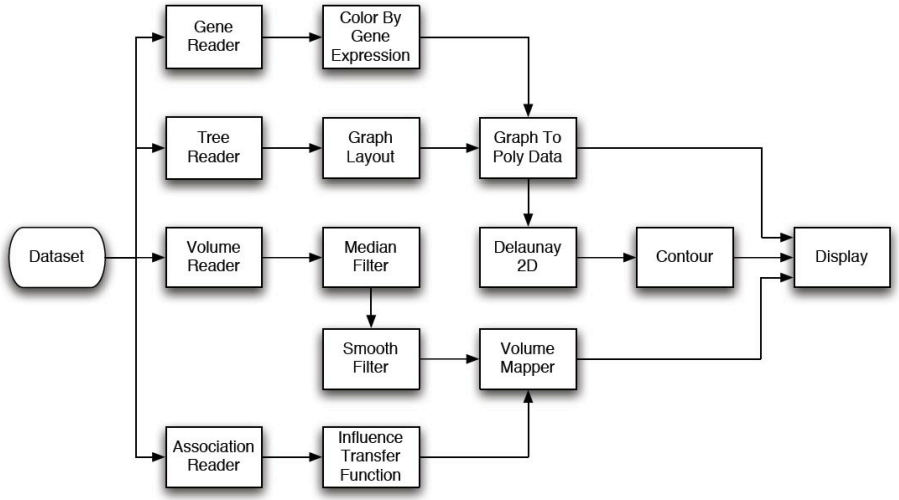


- Application programs written in terms of high-level operations on data
- Runtime system controls scheduling, load balancing, …

2. High-level data-oriented programming model

- Visualization architectures are programmable
  - Uses a data-flow program graph…
- Visualization architectures provide their own run-time system

- Optimize access to numerically-intensive architecture
  - Multi-resolution out-of-core data visualization



Benefiting from a programming model abstraction – visualizing large data

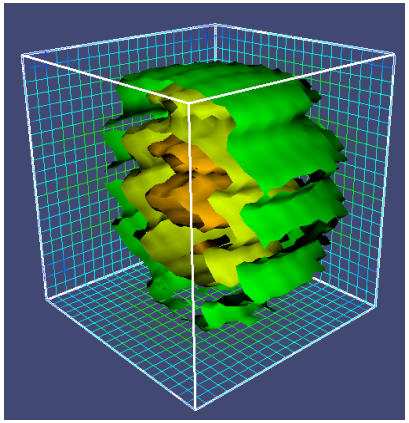
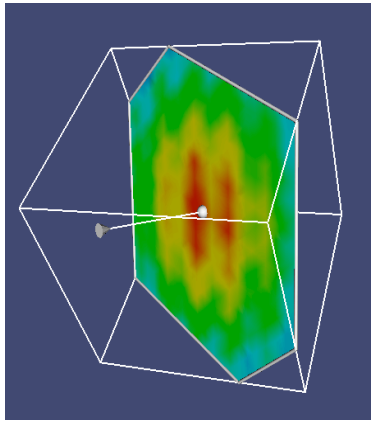
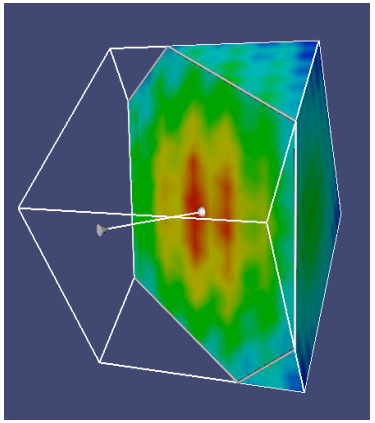
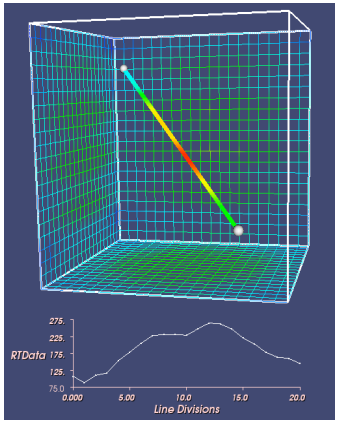
- A decade ago - Large scale data, no visualization solutions
- Los Alamos/Ahrens led project to go:
  - From VTK - An open-source object-oriented visualization toolkit - www.vtk.org
  - To Parallel VTK
  - To ParaView - An open-source, scalable visualization application - www.paraview.org

- Key concepts
  - Streaming is the incremental processing the data as pieces
  - Streaming enables parallelism
    - Pieces processed independently
  - Applied to all operations in the toolkit
    - Contouring, cutting, clipping, analysis

Approach - Data reduction and ordering

Each module in the pipeline can cull and prioritize…

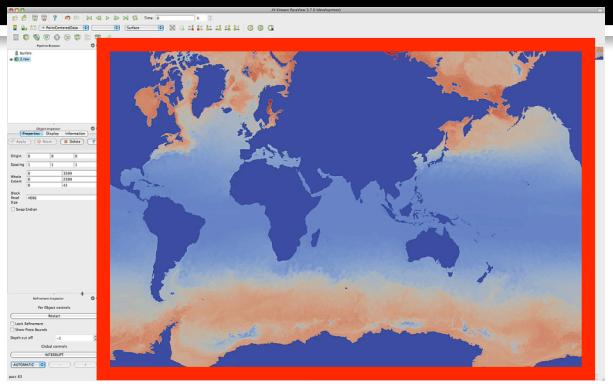
- Culling – remove pieces
  - Based on spatial location
    - Spatial clipping
    - Cutting
    - Probing
    - Frustum culling
    - Occlusion culling
  - Based on data value
    - Contouring
    - Thresholding
- Prioritization – order piece processing
  - Based on spatial location
    - View dependent ordering
  - Based on features
  - Based on user input

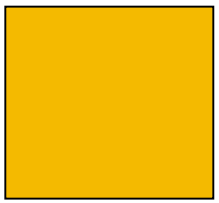





Our current overall visualization approach – enabled by data-oriented programming model

- Data reduction
  - Subsetting the data and culling
  - Sampling the data from disk to create multi-resolution representation
  - Visualization and analysis modules in pipeline – highlighting property of the dataset
    - For example - isosurface, cut plane, clipping
- Prioritization
  - Processing most important data first
- Continuously improve visualized results over time
- Think of a progressive refinement approach of 2D images on the web…
  Our solution provides a prioritized 3D progressive refinement approach that works within a full-featured visualization tool…

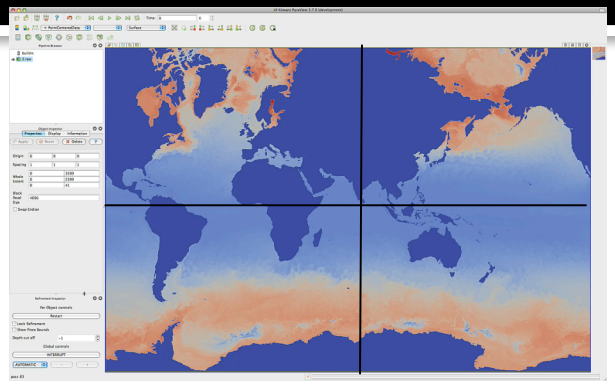
Multi-resolution prioritized streaming





1) Send and render lowest resolution data

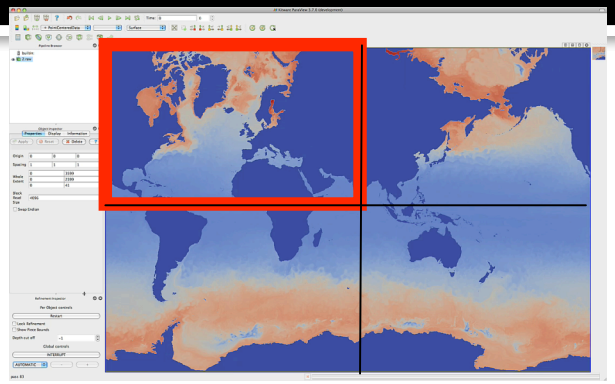
Multi-resolution prioritized streaming

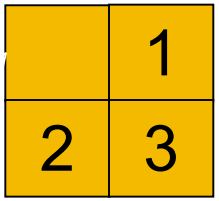


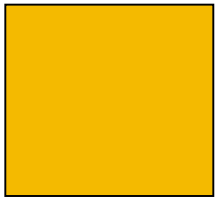
1	2
3	4

1) Send and render lowest resolution data
2) Virtually split into spatial pieces and prioritize pieces

Multi-resolution prioritized streaming

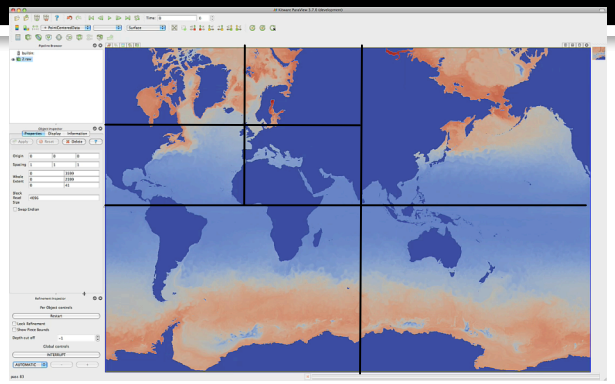






1) Send and render lowest resolution data
2) Virtually split into spatial pieces and prioritize pieces
3) Send and render highest priority piece at higher resolution

Multi-resolution prioritized streaming

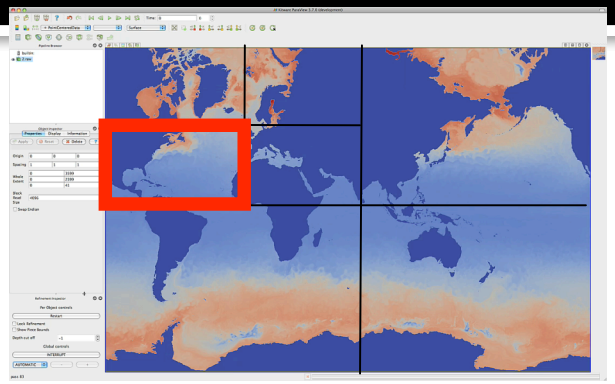


	5
6	7

3	4
1	2

1) Send and render lowest resolution data
2) Virtually split into spatial pieces and prioritize pieces
3) Send and render highest priority piece at higher resolution
4) Goto step 2 until the data is at the highest resolution

Multi-resolution prioritized streaming



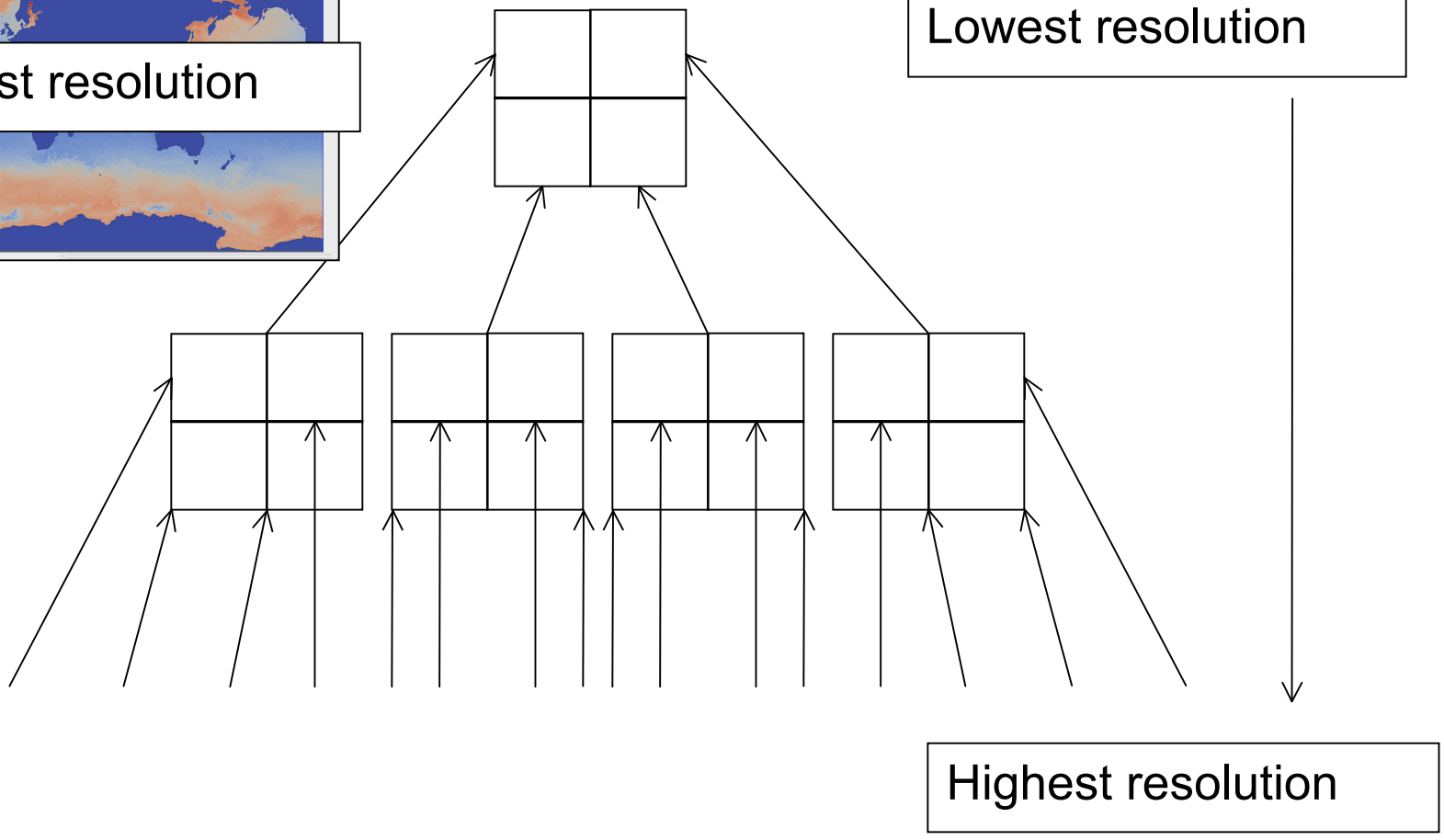
	4
5	6

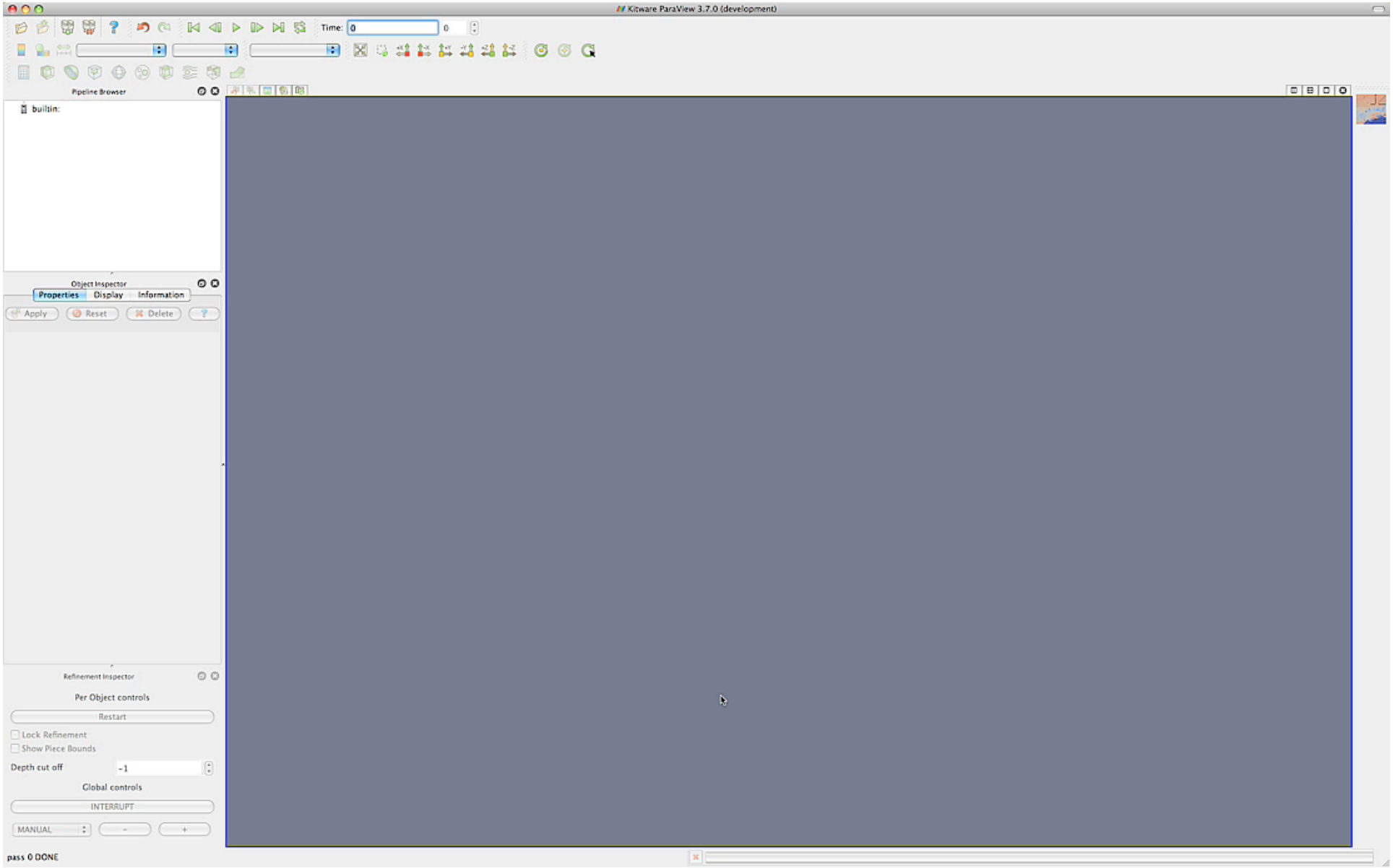
2	3
	1

1) Send and render lowest resolution data
2) Virtually split into spatial pieces and prioritize pieces
3) Send and render highest priority piece at higher resolution
4) Goto step 2 until the data is at the highest resolution

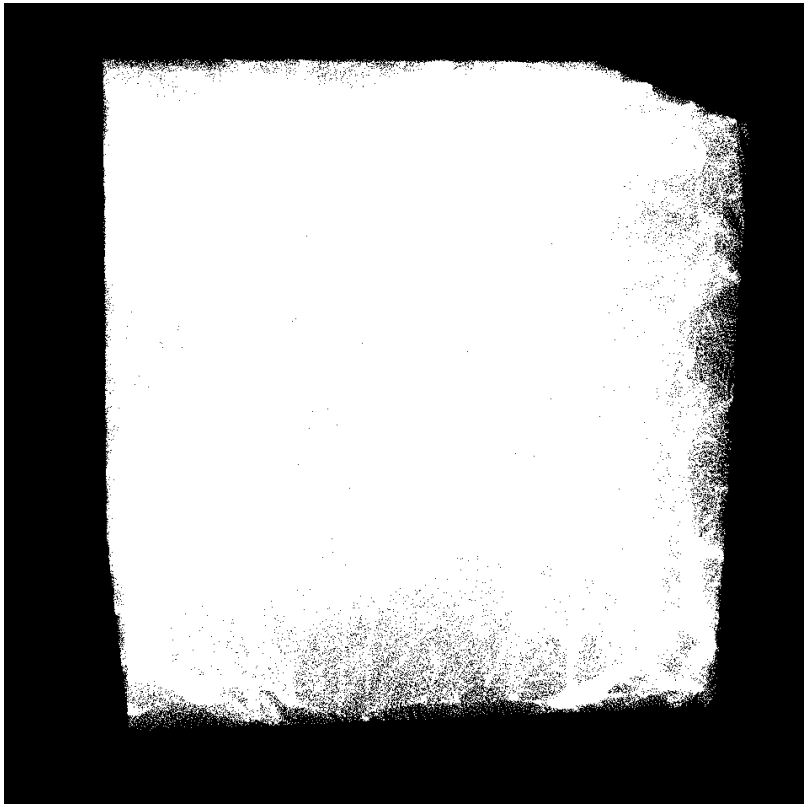
Multi-resolution prioritized streaming

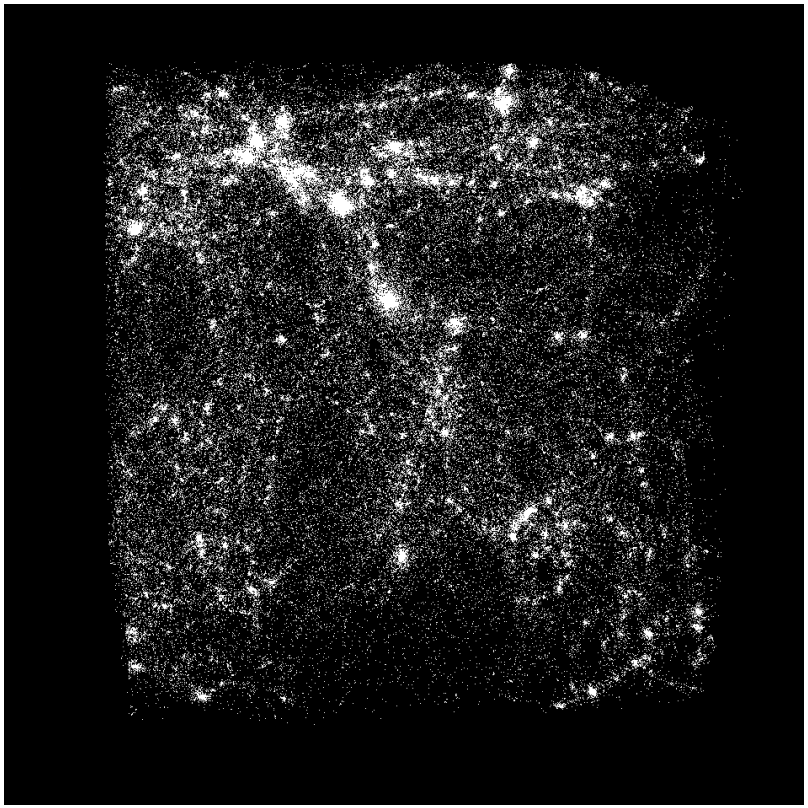
Highest resolution




Kitware ParaView 3.7.0 (development)
Time:
0
Pipeline Browser
builtin:
Object Inspector
Properties
Display
Information
Apply
Reset
Delete
Refinement Inspector
Per Object controls
Restart
Lock Refinement
Show Piece Bounds
Depth cut off
-1
Global controls
INTERRUPT
MANUAL
pass 0 DONE

Multi-Resolution Sampling



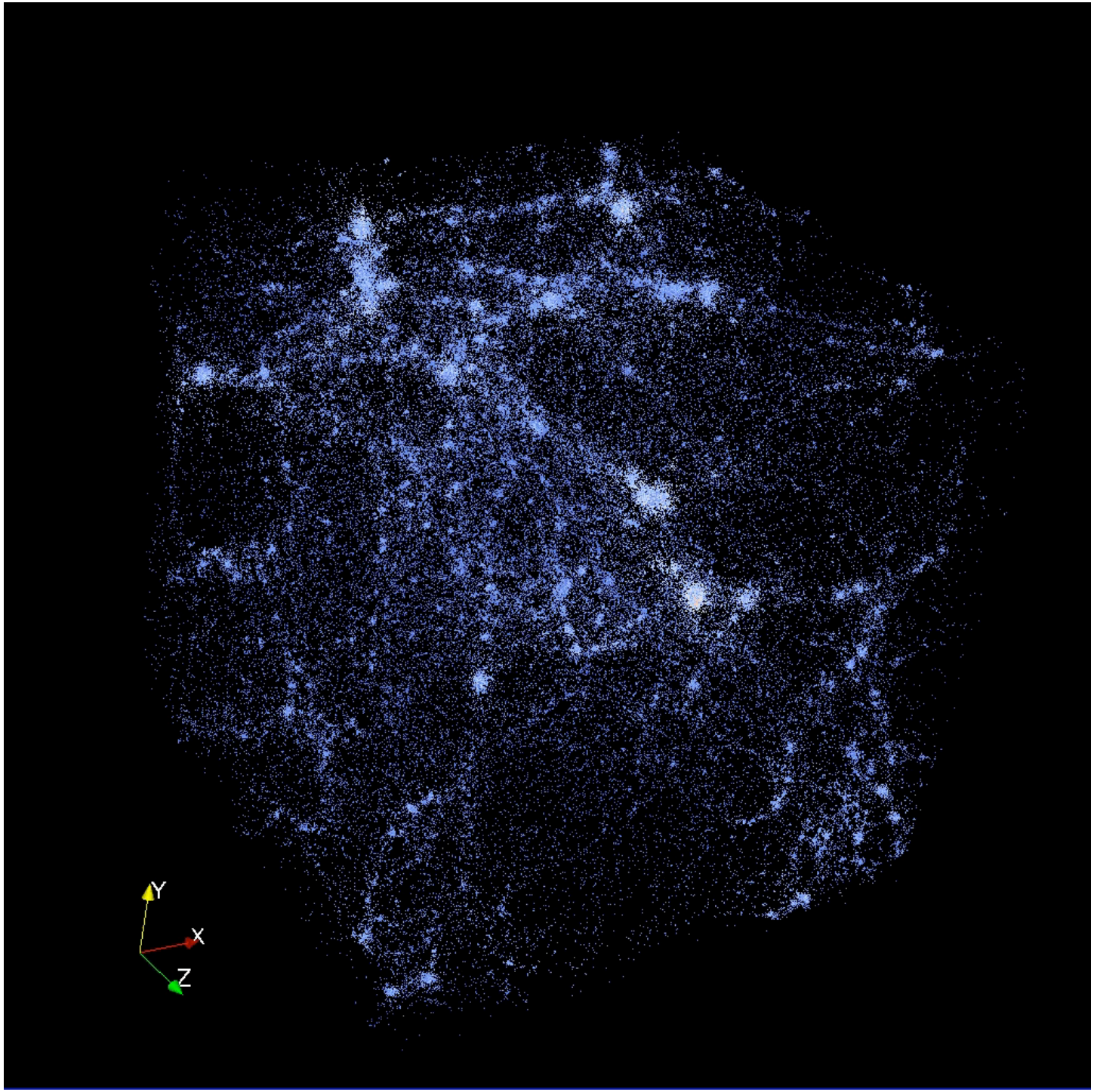


Sampling

- In-situ & storage-based sampling-based data reduction
  - Can work with all data types (structured, unstructured, particle) and most algorithms with little modification
- Intelligent sampling designs to provide more information in less data
  - Little or no processing with simpler sampling strategies (e.g., pure random)
- Untransformed data with error bounds
  - Data in the raw; Ease concerns on unknown transformations/alterations
  - Probabilistic data source as a first-class citizen in visualization and analysis

Sampling

- Quantify the data error for analysis, quantify visual error for vis
  - Show the data error, allow the user to reduce error incrementally
  - Scientist is always informed of the error in their current view
- Data size scales with sample size for bottlenecks
  - Any sample sizes based on error constraints and system/human constraints
  - Same model could be used in simulations to reduce data output per time step


Y
X
Z

3. DISC on numerically-intensive supercomputing platform – interactive access

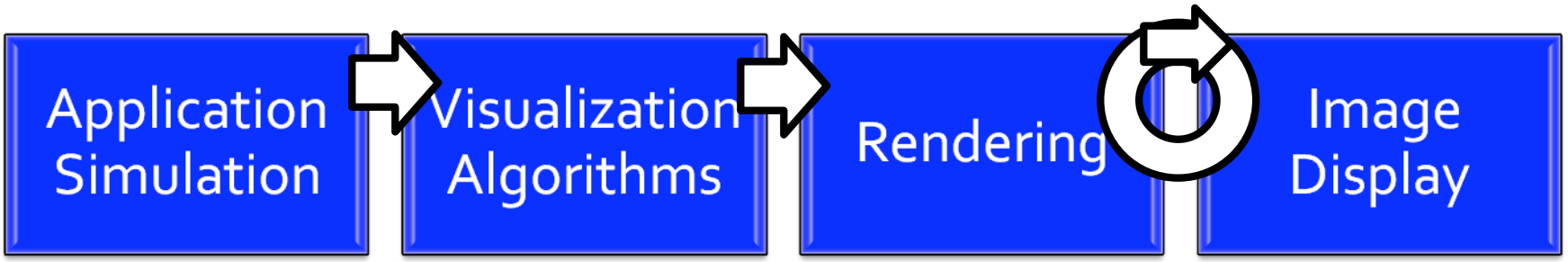
NUMERICALLY INTENSIVE

- Main Machine: Batch Access
  - Priority is to conserve machine resources
  - User submits job with specific resource requirements
  - Run in batch mode when resources available
- Offline Visualization
  - Move results to separate facility for interactive use

DISC

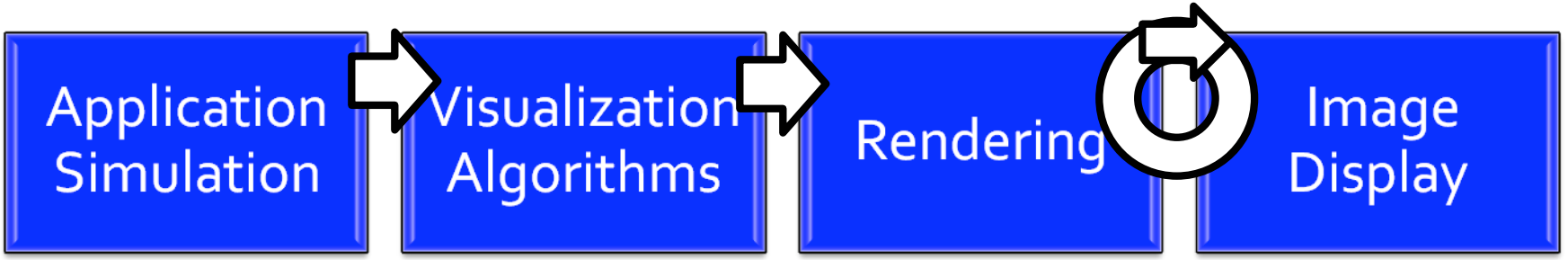
- Interactive Access
  - Priority is to conserve human resources
  - User action can range from simple query to complex computation
  - System supports many simultaneous users
    - Requires flexible programming and runtime environment

Visualization process



Visualization process

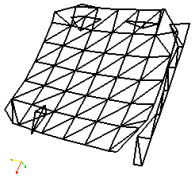
Simulation
Results

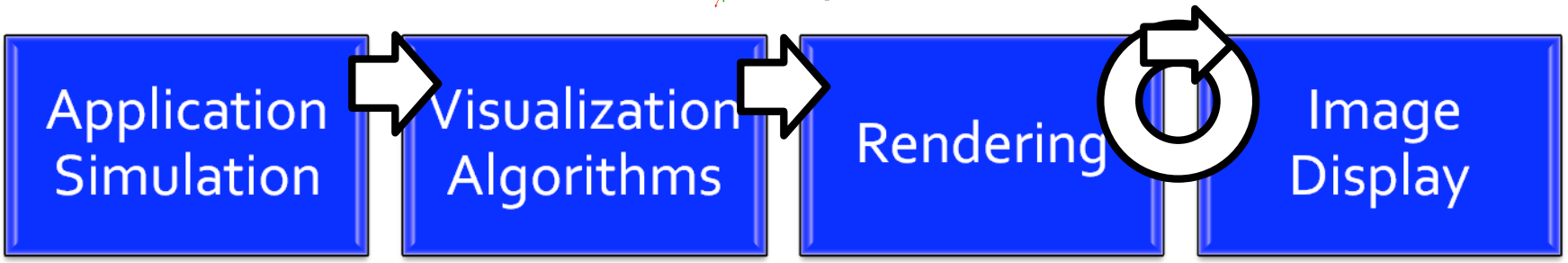


Visualization process

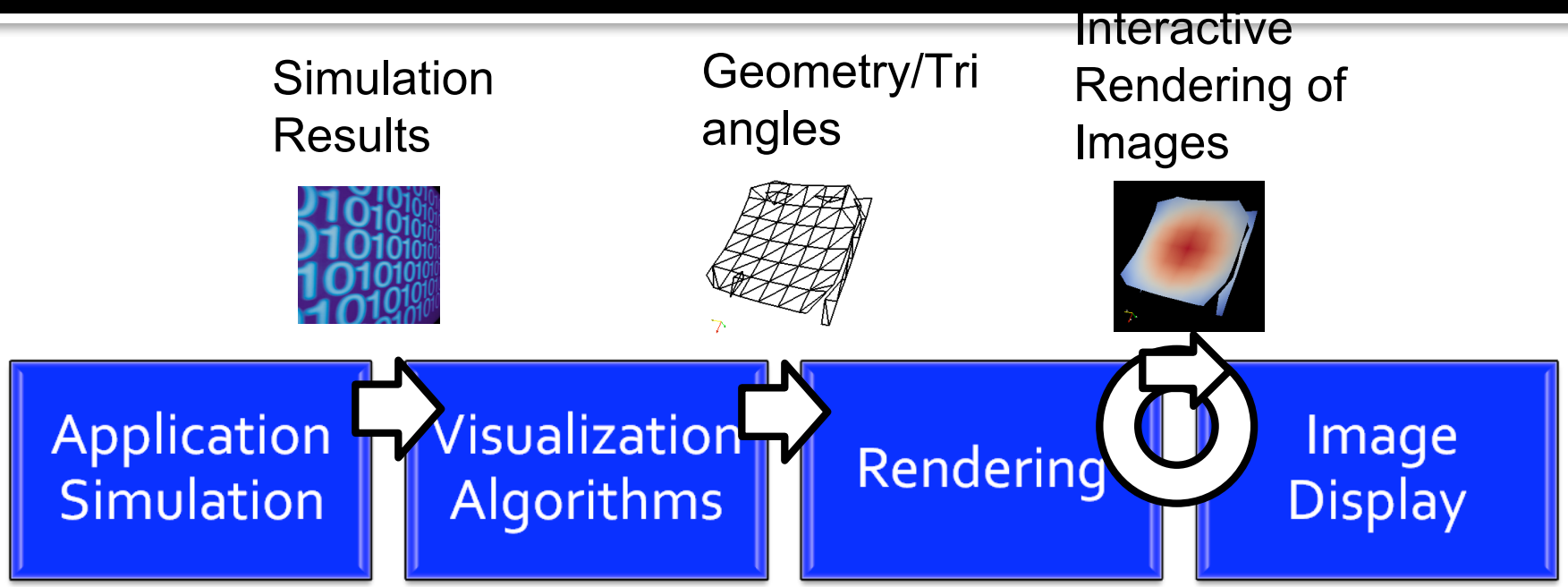
Simulation Results

Geometry/Triangles

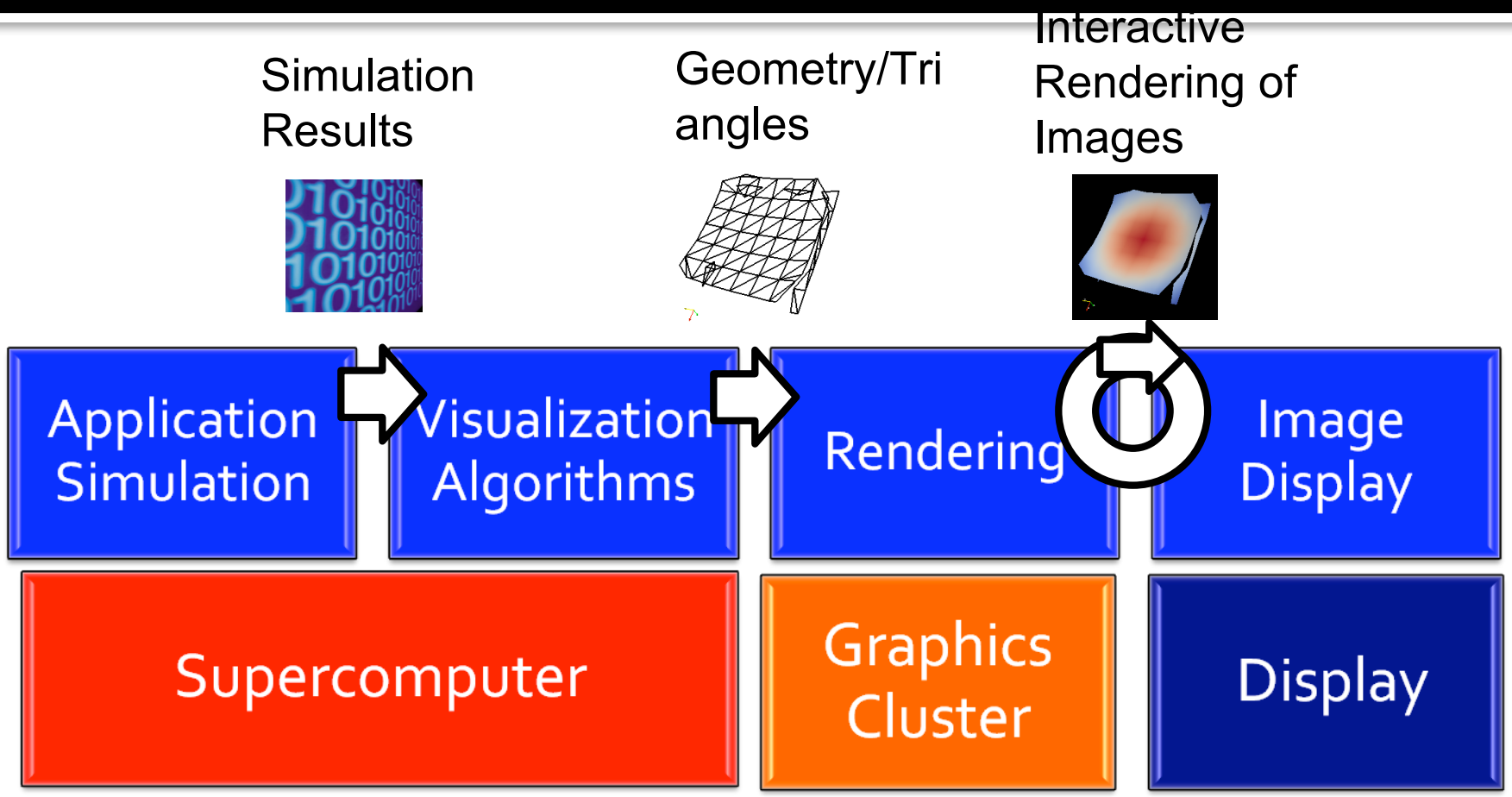




Visualization process



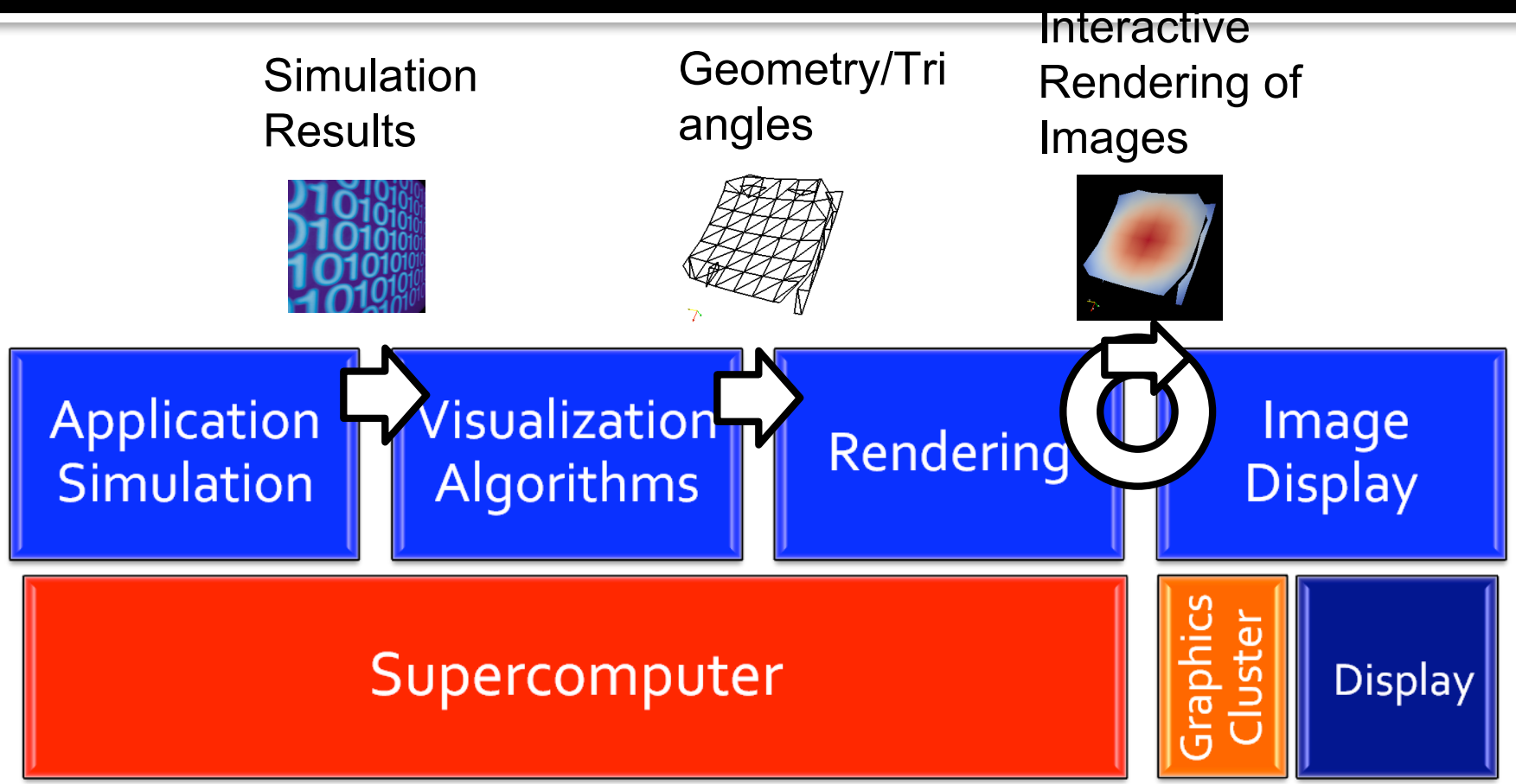
Visualization process mapped on hardware - Current LANL approach



One issue we are evaluating is interactive rendering

- Interactivity is critically important for insight
  - 5-10 fps minimum, 24-30 fps – HDTV, 60 fps – stereo
- There is a cost to achieve interactivity
  - High-performance requirements…
  - Provided by GPU in graphics cluster
  - Can it be provided by the SC platform?

Visualization process mapped on hardware – Alternative approach we are evaluating

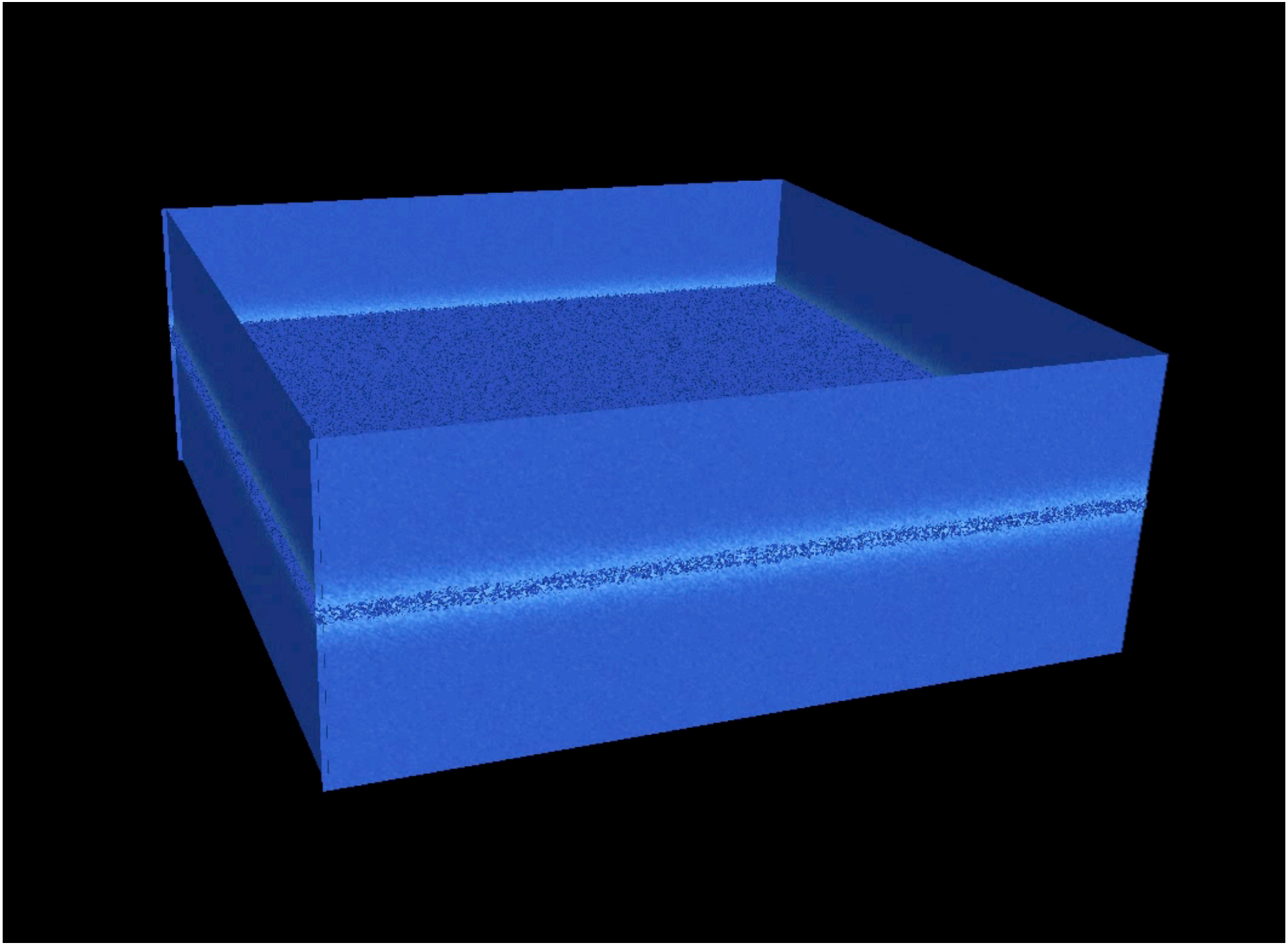


Why is this important?

- Focus is on rendering scalability for exascale datasets
- Our work helps the community to understand:
  - Algorithmic rendering choices for large datasets
  - Architectural rendering choices for large datasets
- Benefits of each CPU/GPU approaches
  - Rendering on supercomputer
    - Scalable using full platform, larger memory
  - Rendering on separate visualization cluster
    - Needed for stereo rendering and displays
    - Useful from a practical perspective
      - Independent resource devoted to visualization tasks

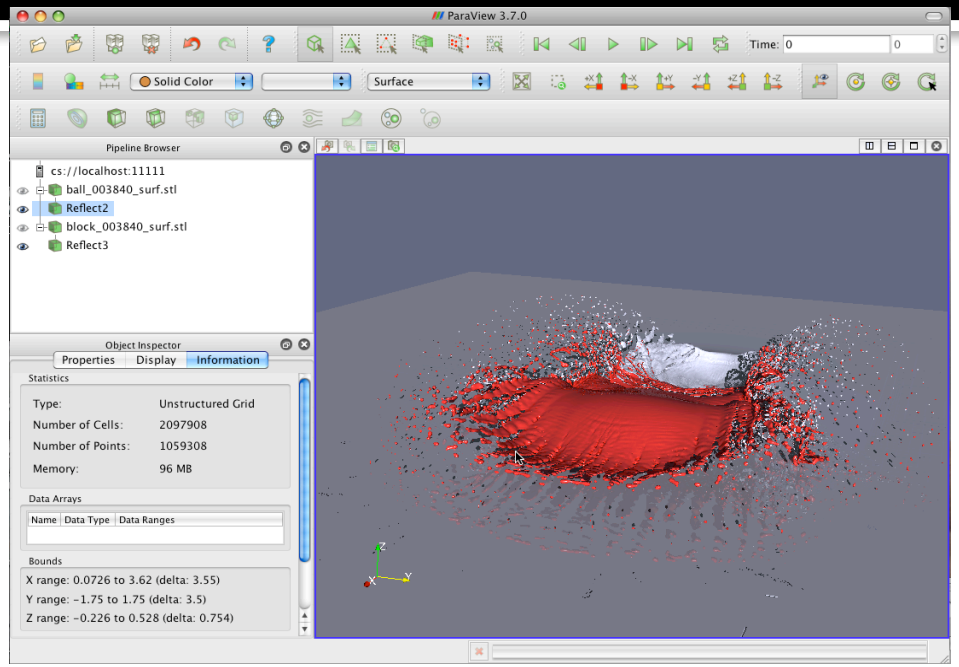
Real world application: VPIC – A case study of interactive visualization on the RoadRunner platform

- Running simulation on 4096 RR processors
  - Computing a 8096x8096x448 grid
- The VPIC team ran their visualization on 128 RR processors
  - Striding and subsetting data to explore and understand their data
- The VPIC team considers interactive visualization critical to the success of their project
  - Bill Daughton, Brian Albright



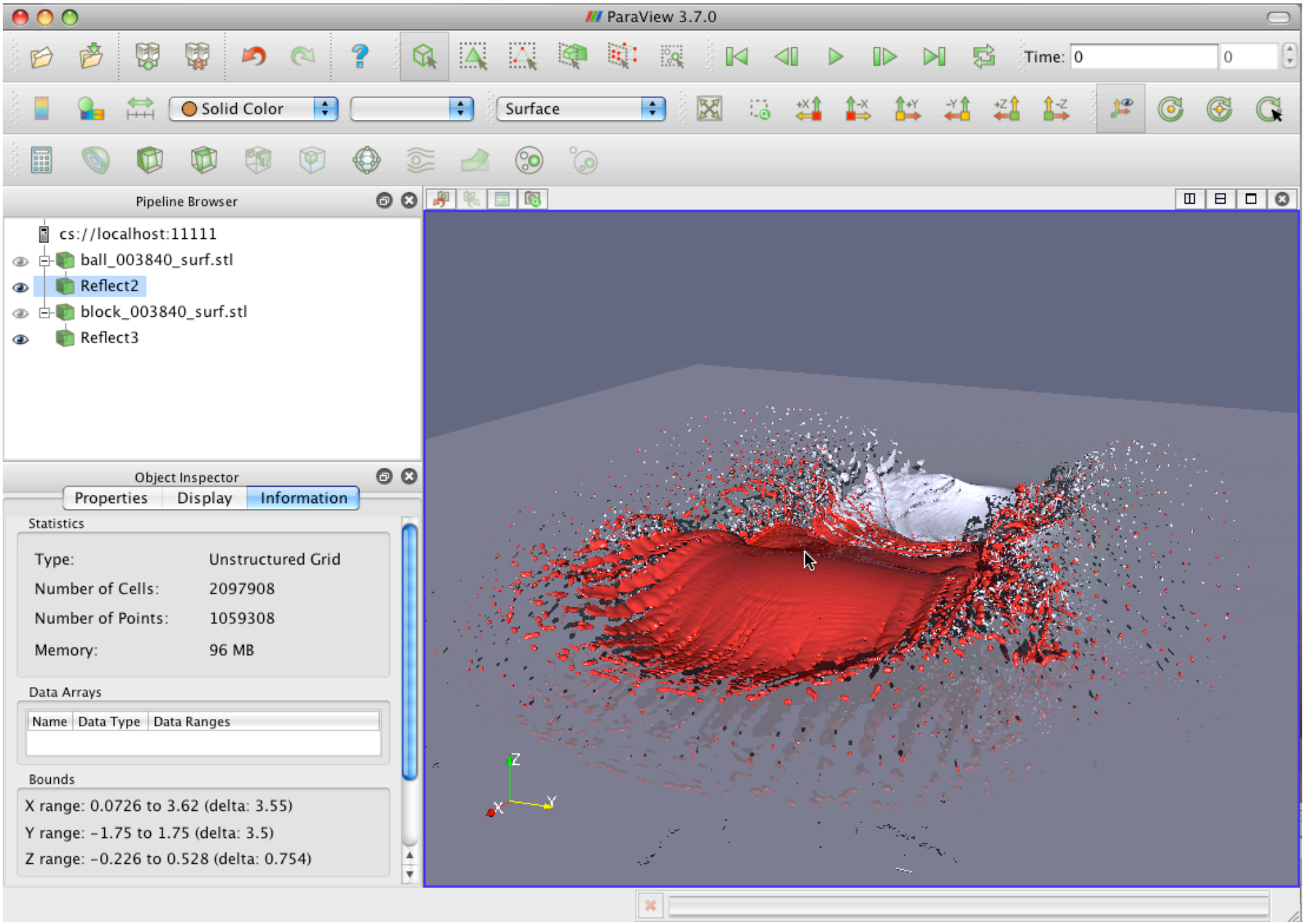
Advance the Maturity Level of CPU Based Rendering on the Supercomputing Platform

- ParaView 3.8.0 released with the Manta Ray Tracer
- Faster than standard software rendering (Mesa)

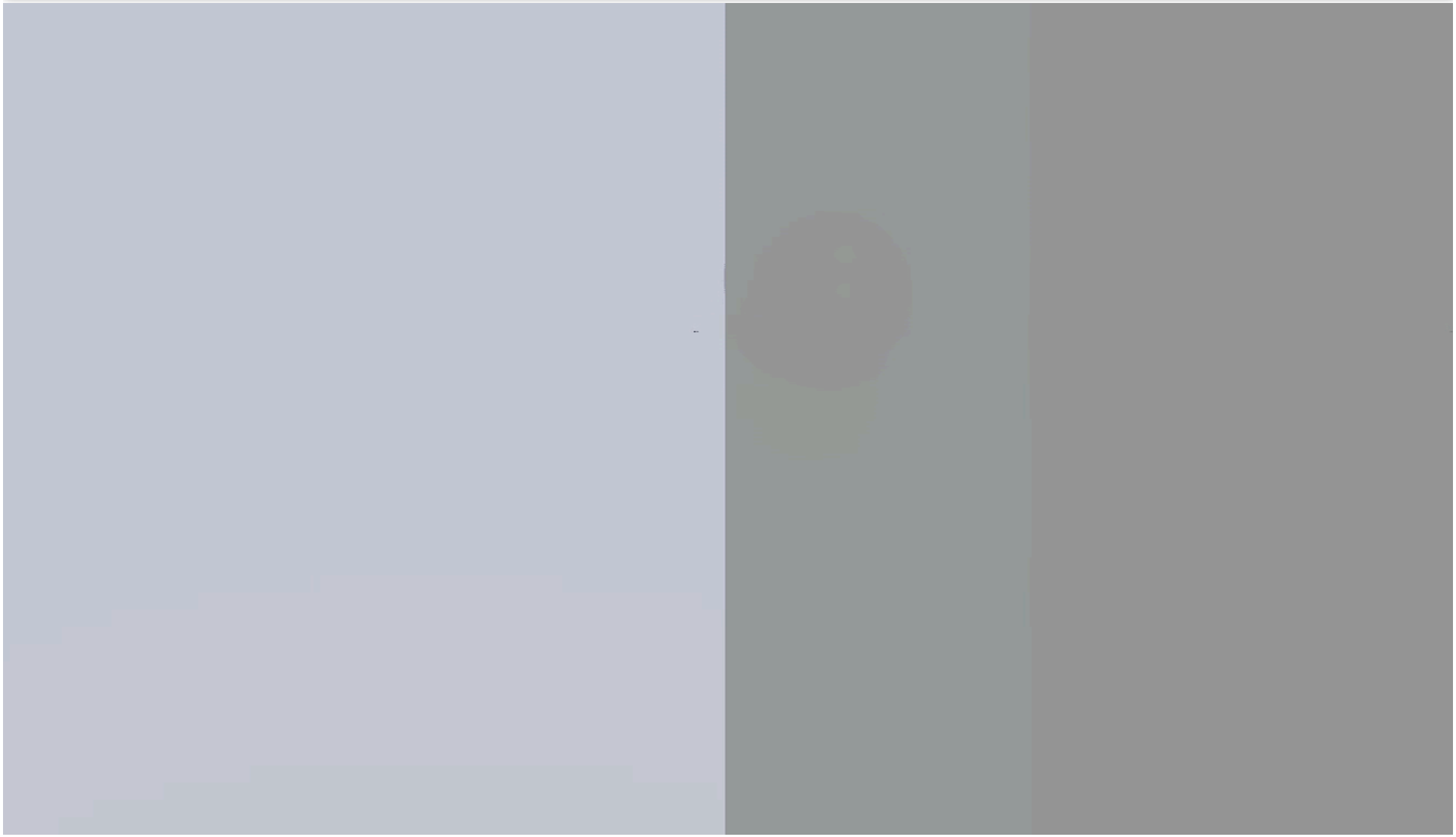


Manta Interactive Ray Tracer

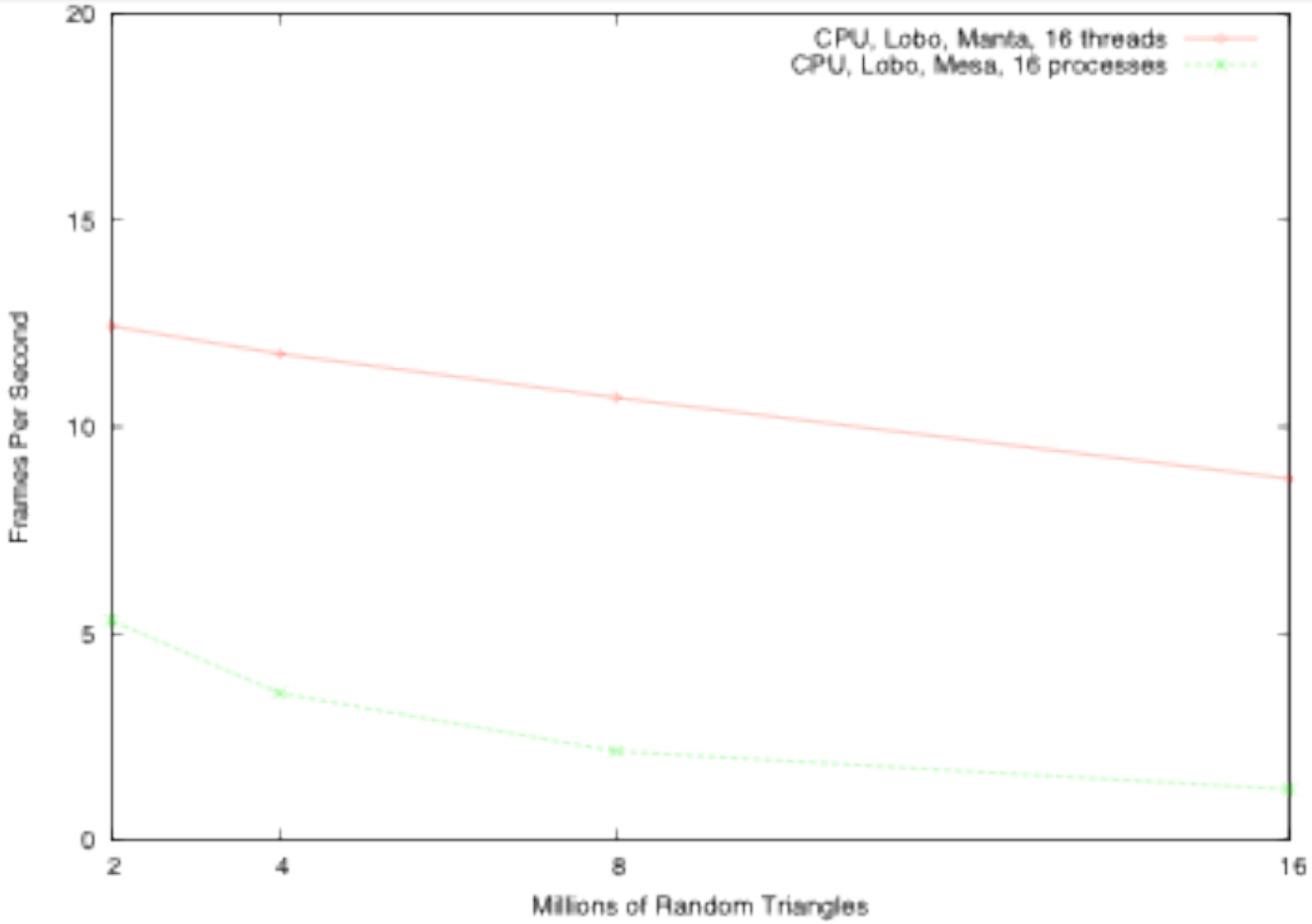
- Fast ray-tracing based renderer
  - Use case 1:
    - Output same results as standard OpenGL renderer
    - What we will use for parallel rendering evaluation tests
  - Use case 2:
    - On a shared-memory machine
      - Shadows and reflections
- Open source development at the University of Utah
- Optimized for multi-core processors
  - Intelligently processes packets of rays together to increase memory locality


ParaView 3.7.0
Time: 0
0
Solid Color
Surface
Pipeline Browser
cs://localhost:11111
ball_003840_surf.stl
Reflect2
block_003840_surf.stl
Reflect3
Object Inspector
Properties
Display
Information
Statistics
Type: Unstructured Grid
Number of Cells: 2097908
Number of Points: 1059308
Memory: 96 MB
Data Arrays
Name
Data Type
Data Ranges
Bounds
X range: 0.0726 to 3.62 (delta: 3.55)
Y range: -1.75 to 1.75 (delta: 3.5)
Z range: -0.226 to 0.528 (delta: 0.754)
Z
X
Y

Use Case: High Fidelity Rendering on Large Shared Memory Machines



CPU Rendering Performance Improved 2-10X with Manta versus Mesa



Evaluate CPU and GPU-based Rendering Performance

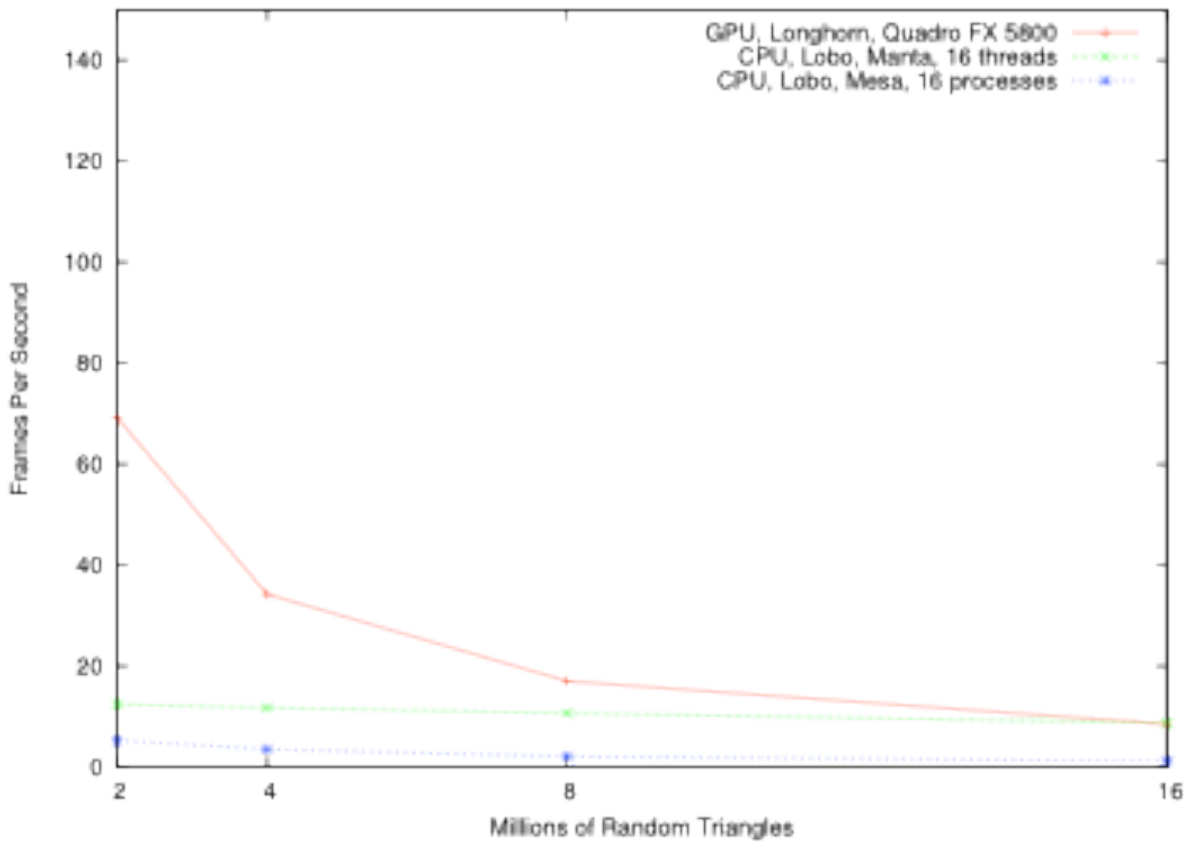
- Machine architectures
  - Lobo – TLCC/CPU cluster
  - Longhorn – GPU cluster
  - Kratos – next gen. CPU cluster
- Datasets - Wavelet, Random Triangles, VPIC
- Number of Polygons/Triangles
  - 0 to 2 billion triangles
    - 1,2,4,8, 16, …, 2048
- Renderers
  - Mesa, Manta, NVidia GPU
- Render Window size
  - 1024 x 1024

Rendering Algorithms

- Scan-conversion of polygons
  - Iterate through all polygons and draw pixels on the image
  - This is the standard rendering approach
    - GPUs, Mesa
  - **~Run time**
    - **O(POLYGONS)**

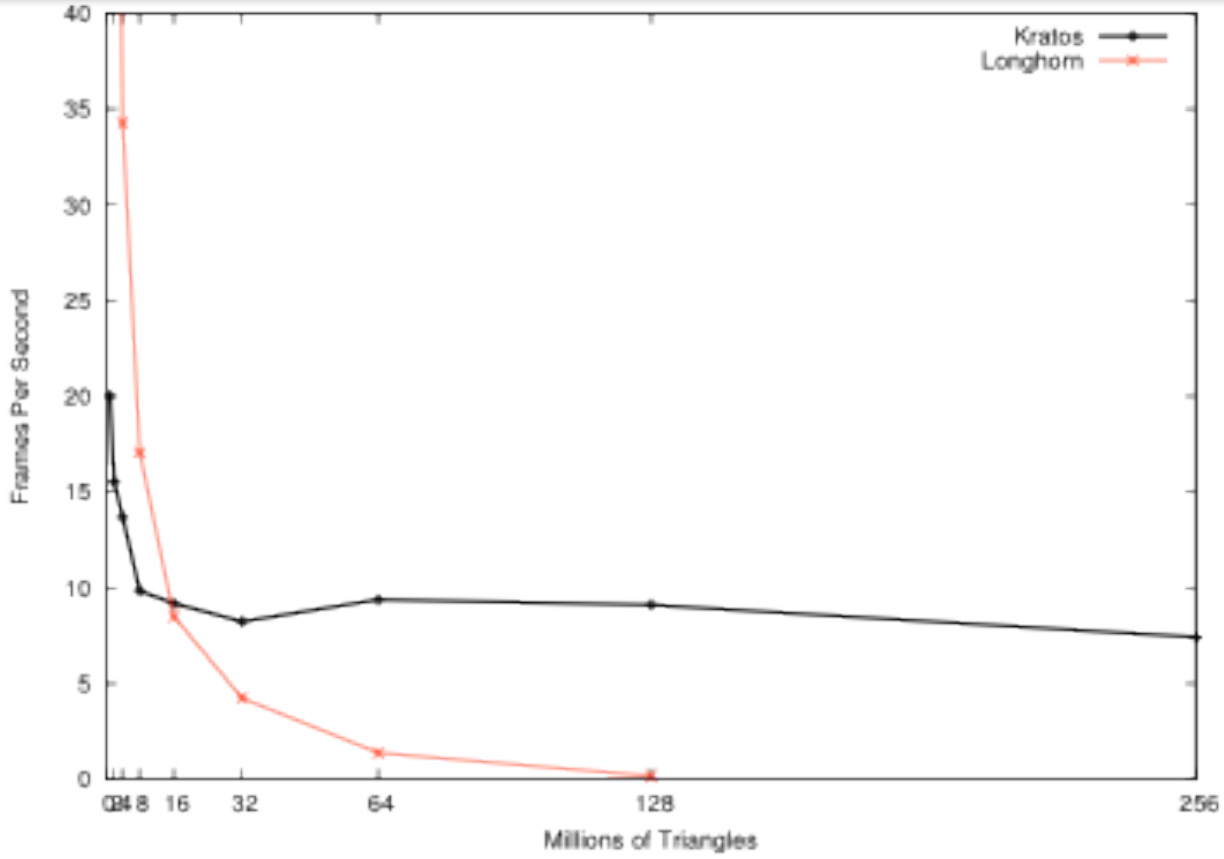
- Ray-tracing
  - Project rays from image into polygon database to compute intersections
  - Tree-based polygon lookup structure
  - **~Run time**
    - **O(IMAGE_SIZE * log (POLYGONS))**

Single Node Rendering Performance – TLCC cluster node vs. GPU cluster node



- GPU is very fast with small polygon counts
  - Important for stereo rendering and tiled displays
- GPU has similar performance to CPU with 16 million triangles

Single Node Rendering Performance
Next Gen. CPU vs. GPU (Large polygon counts)



Single Node Rendering Performance Summary

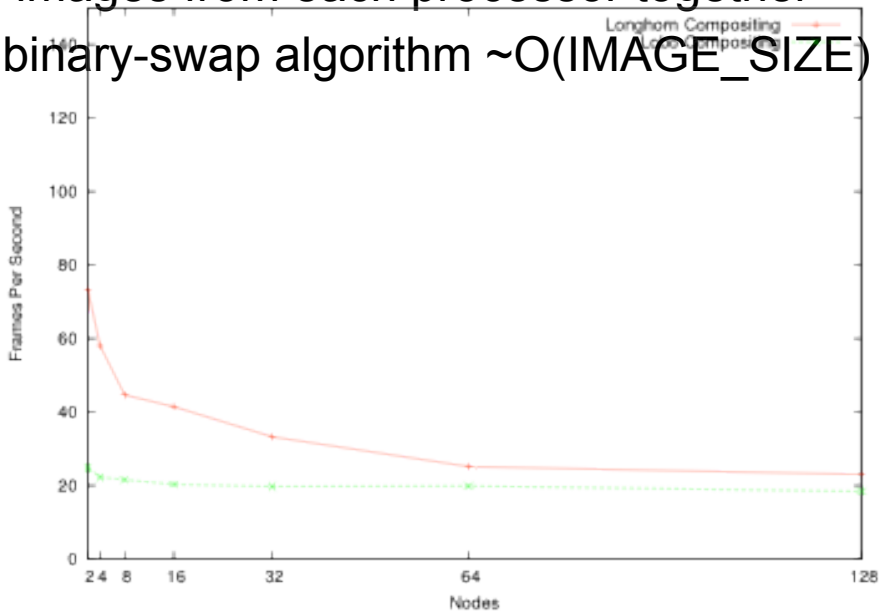
- At 16 million triangles rendering performance of current TLCC cluster node is equal to that of the GPU cluster through ParaView
- As the number of polygons increases:
  - ~stable Manta performance (5-10 FPS) due to:
    - CPU/Manta run time
      - ~O(IMAGE_SIZE * log (POLYGONS))
  - And decreasing GPU performance due to:
    - GPU run time
      - ~O(POLYGONS)

Efficient Rendering Algorithms for Exascale

- As we move towards exascale -- polygon counts will increase
  - We would prefer rendering algorithms that are image size dependent instead of polygon count dependent
- We see this as a rendering algorithm issue
  - Hardware acceleration possible:
    - GPU vendors could implement algorithms/data structures that support large polygons counts
      - NVidia ray-tracing Optix library

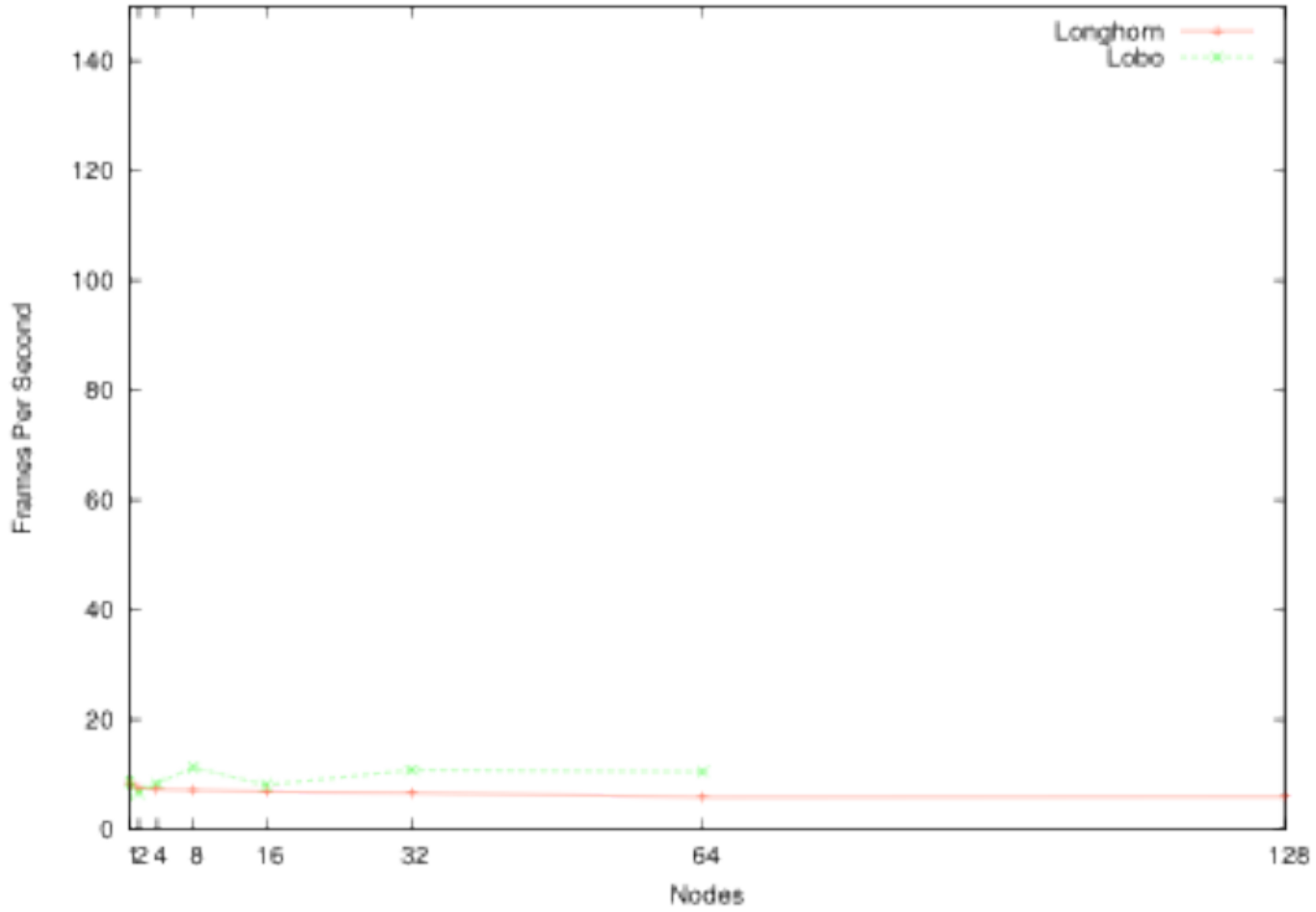
Parallel rendering approaches

- Total Time = Rendering time + Compositing time
  - Rendering
    - Locally, draw image from geometry on each processor
  - Compositing communication step
    - Merge images from each processor together
    - Using binary-swap algorithm ~O(IMAGE_SIZE)



Parallel Rendering Performance for Weak Scaling – TLCC cluster vs. GPU cluster

16 Million Triangles on each Node



Parallel Rendering Performance for Weak Scaling

- Recall, rendering 16 million polygons at similar rates with both CPU and GPU resources
  - We expect to see a relatively similar constant graphs
    - Rendering cost for 16 million triangles on each node out weighs compositing costs
  - 128 nodes * 16 million triangles/node =
    - 2,048 million triangles total
    - 2,048 “mega” triangles
    - 2 “giga” triangles

Parallel Rendering Performance for Weak Scaling

- We don't have a cluster of next generation machines yet, however… (review perf. – next slide)
- Rendering 256 million polygons per node with CPU and GPU resources
  - We would expect to see constant graphs
    - CPU/Manta performance would be at ~10 FPS
      - This is the scalable performance we need
    - GPU performance less than ~1FPS
  - Could render -- 128 nodes * 256 million triangles/node
    - 32768 million triangles total
    - 32 "Giga" triangles

Conclusions

- New data-intensive approaches needed for exascale
  - In-situ/feature extraction
  - Data sampling
  - Rendering on the platform